Announcements

• Homework 1 due **today (Wednesday) at 8pm**

• Quiz 1 due **tomorrow (Thursday) at 8pm**

• **Project:** Links to past projects, milestone templates posted

• Homework 2, Quiz 2 will be released tonight
  • Covers linear and logistic regression
  • **HW 2 has a slightly extended deadline (Monday, October 3 at 8pm)**
Recap: $L_2$ Regularization

- **Original MSE loss + regularization:**

\[
L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \cdot \|\beta\|_2^2
\]

- $\lambda$ is a **hyperparameter** that must be tuned (satisfies $\lambda \geq 0$)
Recap: $L_2$ Regularization
Recap: $L_2$ Regularization

At this point, the gradients are equal (with opposite sign).

Tradeoff depends on choice of $\lambda$.

- Loss varies greatly in this direction.
- Penalizes more.

Minimizes original loss (or if $\lambda = 0$).

Minimizes full loss.

Minimizes regularization term (or if $\lambda \to \infty$).

\[
L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2
\]
Recap: $L_1$ Regularization

\[
L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{d} |\beta_j|
\]

Minimizes original loss (or if $\lambda = 0$)

Minimizer of full loss at corner $\rightarrow$ sparse ($\beta_1 = 0$)

Minimizes regularization term (or if $\lambda \rightarrow \infty$)
Recap: $L_1$ Regularization

• **Step 1:** Construct a lot of features and add to feature map

• **Step 2:** Use $L_1$ regularized regression to “select” subset of features
  • I.e., coefficient $\beta_j \neq 0 \rightarrow$ feature $j$ is selected
  • Tune $\lambda$ to select more/fewer features

• **Optional:** Remove unselected features from the feature map and run vanilla linear regression (a.k.a. ordinary least squares)
Recap: Cross Validation

• **Original MSE loss + regularization:**

\[
L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 + \lambda \cdot \|\beta\|_2^2
\]

• \(\lambda\) is a **hyperparameter** that must be tuned (satisfies \(\lambda \geq 0\))

• How to choose \(\lambda\)?
Recap: Cross Validation

<table>
<thead>
<tr>
<th>Training data $Z_{\text{train}}$</th>
<th>Val data $Z_{\text{val}}$</th>
<th>Test data $Z_{\text{test}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1 = 0.01$</td>
<td>$\hat{\beta}<em>1 \leftarrow \beta(Z</em>{\text{train}}, \lambda_1)$</td>
<td>$L^1_{\text{val}} \leftarrow L(\hat{\beta}<em>1; Z</em>{\text{val}})$</td>
</tr>
<tr>
<td>$\lambda_2 = 0.10$</td>
<td>$\hat{\beta}<em>2 \leftarrow \beta(Z</em>{\text{train}}, \lambda_2)$</td>
<td>$L^2_{\text{val}} \leftarrow L(\hat{\beta}<em>2; Z</em>{\text{val}})$ $L(\hat{\beta}<em>{t'}; Z</em>{\text{test}})$</td>
</tr>
<tr>
<td>$\lambda_2 = 1.00$</td>
<td>$\hat{\beta}<em>3 \leftarrow \beta(Z</em>{\text{train}}, \lambda_3)$</td>
<td>$L^3_{\text{val}} \leftarrow L(\hat{\beta}<em>3; Z</em>{\text{val}})$</td>
</tr>
</tbody>
</table>

$t' \leftarrow \max_t L^t_{\text{val}}$
Recap: Cross Validation

• Generally important for tuning design choices
  • Hyperparameters
  • Features in the feature map
  • Model family
  • ...

• Alternative approaches exist for very small datasets
  • Re-train on $Z_{\text{train}} \cup Z_{\text{val}}$
  • $k$-fold cross validation
Agenda

• Minimizing the MSE Loss
  • Closed-form solution
  • Gradient descent
Minimizing the MSE Loss

• Recall that linear regression minimizes the loss

\[ L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 \]

• **Closed-form solution**: Compute using matrix operations

• **Optimization-based solution**: Search over candidate \( \beta \)
Recap: Closed-Form Solution

- Minimum solution has gradient equal to zero:

$$\nabla_\beta L(\hat{\beta}; Z) = 0$$
Recap: Closed-Form Solution

\[
\begin{bmatrix}
  f_\beta(x_1) \\
  \vdots \\
  f_\beta(x_n)
\end{bmatrix} = \begin{bmatrix}
  \beta^T x_1 \\
  \vdots \\
  \beta^T x_n
\end{bmatrix} = \begin{bmatrix}
  \sum_{j=1}^{d} \beta_j x_{1,j} \\
  \vdots \\
  \sum_{j=1}^{d} \beta_j x_{n,j}
\end{bmatrix} = \begin{bmatrix}
  x_{1,1} & \cdots & x_{1,d} \\
  \vdots & \ddots & \vdots \\
  x_{n,1} & \cdots & x_{n,d}
\end{bmatrix} \begin{bmatrix}
  \beta_1 \\
  \vdots \\
  \beta_d
\end{bmatrix} = X\beta
\]

\[\therefore \]

\[
\begin{bmatrix}
  y_1 \\
  \vdots \\
  y_n
\end{bmatrix} = Y
\]

\textbf{Summary: } Y \approx X\beta
Recap: Closed-Form Solution

\[ L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2 = \frac{1}{n} \|Y - X\beta\|_2^2 \]

\[ \|z\|_2^2 = \sum_{i=1}^{n} z_i^2 \]
Recap: Closed-Form Solution

• Minimizer of the MSE loss has gradient equal to zero:

\[ \nabla_{\beta} L(\hat{\beta}; Z) = 0 \]
Recap: Closed-Form Solution

- The gradient is

\[
\nabla_{\beta} L(\beta; Z) = \nabla_{\beta} \frac{1}{n} \|Y - X\beta\|_2^2 = \nabla_{\beta} \frac{1}{n} (Y - X\beta)^T (Y - X\beta)
\]

\[
= \frac{2}{n} [\nabla_{\beta} (Y - X\beta)^T] (Y - X\beta)
\]

\[
= -\frac{2}{n} X^T (Y - X\beta)
\]

\[
= -\frac{2}{n} X^T Y + \frac{2}{n} X^T X\beta
\]
Recap: Closed-Form Solution

• The gradient is

\[
\nabla_\beta L(\beta; Z) = \nabla_\beta \frac{1}{n} \| Y - X\beta \|_2^2 = -\frac{2}{n} X^T Y + \frac{2}{n} X^T X\beta
\]

• Setting \( \nabla_\beta L(\hat{\beta}; Z) = 0 \), we have \( X^T X\hat{\beta} = X^T Y \)
Recap: Closed-Form Solution

• Setting $\nabla_{\beta} L(\hat{\beta}; Z) = 0$, we have $X^T X \hat{\beta} = X^T Y$

• Assuming $X^T X$ is invertible, we have

$$\hat{\beta}(Z) = (X^T X)^{-1} X^T Y$$
Shortcomings of Closed-Form Solution

• Computing $\hat{\beta}(Z) = (X^T X)^{-1} X^T Y$ can be challenging when the number of features $d$ is large

• Computing $(X^T X)^{-1}$ is $O(d^3)$
  • $d = 10^4$ features $\rightarrow O(10^{12})$
  • Even storing $X^T X$ requires a lot of memory

• Numerical accuracy issues due to “ill-conditioning”
  • What if $X^T X$ is “barely” invertible?
  • Then, $(X^T X)^{-1}$ has large variance along some dimension
  • Regularization helps (more on this later)
Optimization Algorithms

• Recall that linear regression minimizes the loss

\[ L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 \]

• Iteratively optimize \( \beta \)
  • Initialize \( \beta_1 \leftarrow \text{Init}(...) \)
  • For some number of iterations \( T \), update \( \beta_t \leftarrow \text{Step}(...) \)
  • Return \( \beta_T \)
Optimization Algorithms

• **Global search**: Try random values of $\beta$ and choose the best
  • I.e., $\beta_t$ independent of $\beta_{t-1}$
  • Very unstructured, can take a long time (especially in high dimension $d$)!

• **Local search**: Start from some initial $\beta$ and make local changes
  • I.e., $\beta_t$ is computed based on $\beta_{t-1}$
  • What is a “local change”, and how do we find good one?
Strategy 2: Gradient Descent

- **Gradient descent**: Update $\beta$ based on gradient $\nabla_{\beta} L(\beta; Z)$ of $L(\beta; Z)$:

  $$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_{\beta} L(\beta_t; Z)$$

- **Intuition**: The gradient is the direction along which $L(\beta; Z)$ changes most quickly as a function of $\beta$

- $\alpha \in \mathbb{R}$ is a hyperparameter called the **learning rate**
  
  - More on this later
Strategy 2: Gradient Descent

• Choose initial value for $\beta$
• Until we reach a minimum:
  • Choose a new value for $\beta$ to reduce $L(\beta; Z)$
Strategy 2: Gradient Descent

• Choose initial value for $\beta$
• Until we reach a minimum:
  • Choose a new value for $\beta$ to reduce $L(\beta; Z)$

$L(\beta; Z)$

Figure by Andrew Ng
Strategy 2: Gradient Descent

• Choose initial value for $\beta$
• Until we reach a minimum:
  • Choose a new value for $\beta$ to reduce $L(\beta; Z)$

Linear regression loss is convex, so no local minima
Strategy 2: Gradient Descent

• Initialize $\beta_1 = 0$
• Repeat until convergence:

$$
\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_\beta L(\beta_t; Z)
$$

• For linear regression, know the gradient from strategy 1

For in-place updates $\beta \leftarrow \beta - \alpha \cdot \nabla_\beta L(\beta; Z)$, compute all components of $\nabla_\beta L(\beta; Z)$ before modifying $\beta$. 
Strategy 2: Gradient Descent

• Initialize $\beta_1 = 0$
• Repeat until convergence:

$$
\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_{\beta} L(\beta_t; Z)
$$

• For linear regression, know the gradient from strategy 1
Strategy 2: Gradient Descent

• Initialize $\beta_1 = \vec{0}$
• Repeat until $\|\beta_t - \beta_{t+1}\|_2 \leq \epsilon$

$$\beta_{t+1} \leftarrow \beta_t - \alpha \cdot \nabla_{\beta} L(\beta_t; Z)$$

• For linear regression, know the gradient from strategy 1

Hyperparameter defining convergence
Strategy 2: Gradient Descent

\[ h(x) = -900 - 0.1x \]

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
**Strategy 2: Gradient Descent**

\[ f_\beta(x) \quad \text{and} \quad L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

\[ L(\beta; Z) \]
Strategy 2: Gradient Descent

\[ f_\beta(x) \]

Minimizer of loss function

\[ L(\beta; Z) \]
Choice of Learning Rate $\alpha$

$L(\beta; Z)$

**Problem:** $\alpha$ too small
- $L(\beta; Z)$ decreases slowly

**Problem:** $\alpha$ too large
- $L(\beta; Z)$ increases!

Plot $L(\beta_t; Z_{\text{train}})$ vs. $t$ to diagnose these problems
Choice of Learning Rate $\alpha$

• $\alpha$ is a hyperparameter for gradient descent that we need to choose
  • Can set just based on training data

• Rule of thumb
  • $\alpha$ too small: Loss decreases slowly
  • $\alpha$ too large: Loss increases!

• Try rates $\alpha \in \{1.0, 0.1, 0.01, \ldots\}$ (can tune further once one works)
Comparison of Strategies

• **Closed-form solution**
  • No hyperparameters
  • Slow if $n$ or $d$ are large

• **Gradient descent**
  • Need to tune $\alpha$
  • Scales to large $n$ and $d$

• For linear regression, there are better optimization algorithms, but gradient descent is very general
  • Accelerated gradient descent is an important tweak that improves performance in practice (and in theory)
$L_2$ Regularized Linear Regression

- Recall that linear regression with $L_2$ regularization minimizes the loss

$$L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2$$
**$L_2$ Regularized Linear Regression**

- Recall that linear regression with $L_2$ regularization minimizes the loss

$$L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T x_i)^2 + \lambda \sum_{j=1}^{d} \beta_j^2 = \frac{1}{n} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_2^2$$

- Gradient is

$$\nabla_{\beta} L(\beta; Z) = -\frac{2}{n} X^T Y + \frac{2}{n} X^T X \beta + 2 \lambda \beta$$
Strategy 1: Closed-Form Solution

• Gradient is

\[ \nabla_{\beta} L(\beta; Z) = -\frac{2}{n} X^T Y + \frac{2}{n} X^T X \beta + 2\lambda \beta \]

• Setting \( \nabla_{\beta} L(\hat{\beta}; Z) = 0 \), we have \((X^T X + n\lambda I)\hat{\beta} = X^T Y\)

• Always invertible if \( \lambda > 0 \), so we have

\[ \hat{\beta}(Z) = (X^T X + n\lambda I)^{-1} X^T Y \]
Strategy 2: Gradient Descent

• Gradient is

\[
\nabla_{\beta} L(\beta; Z) = -\frac{2}{n}X^T Y + \frac{2}{n}X^T X \beta + 2\lambda \beta
\]

• Same algorithm as vanilla linear regression (a.k.a. OLS)
• **Intuition:** The extra term \( \lambda \beta \) in the gradient is **weight decay** that encourages \( \beta \) to be small
What About $L_1$ Regularization?

• Gradient descent still works!

• Specialized algorithms work better in practice
  • **Simple one:** Gradient descent + soft thresholding
  • Basically, if $|\beta_{t,j}| \leq \lambda$, just set it to zero
  • Good theoretical properties
Loss Minimization View of ML

• Two design decisions
  • Model family: What are the candidate models $f$? (E.g., linear functions)
  • Loss function: How to define “approximating”? (E.g., MSE loss)
Loss Minimization View of ML

• **Three design decisions**
  • **Model family:** What are the candidate models $f$? (E.g., linear functions)
  • **Loss function:** How to define “approximating”? (E.g., MSE loss)
  • **Optimizer:** How do we minimize the loss? (E.g., gradient descent)
Lecture 5: Logistic Regression

CIS 4190/5190
Fall 2022
Supervised Learning

Data $Z = \{(x_i, y_i)\}_{i=1}^n$

$\hat{\beta}(Z) = \arg\min_{\beta} L(\beta; Z)$

$L$ encodes $y_i \approx f_\beta(x_i)$

Model $f_{\hat{\beta}(Z)}$
Regression

Data $Z = \{(x_i, y_i)\}_{i=1}^n$  

$\hat{\beta}(Z) = \arg \min_{\beta} L(\beta; Z)$  
$L$ encodes $y_i \approx f_{\beta}(x_i)$

Model $f_{\hat{\beta}(Z)}$

Label is a **real value** $y_i \in \mathbb{R}$
Classification

Data \( Z = \{(x_i, y_i)\}_{i=1}^n \)  \( \hat{\beta}(Z) = \arg\min_\beta L(\beta; Z) \)  
\( L \) encodes \( y_i \approx f_\beta(x_i) \)

Model \( f_{\hat{\beta}(Z)} \)

Label is a **discrete value** \( y_i \in Y = \{c_1, \ldots, c_k\} \)
(Binary) Classification

- **Input:** Dataset \( Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}\)
- **Output:** Model \( y_i \approx f_\beta(x_i) \)

**Example:** Malignant vs. Benign Ocular Tumor

[Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/]

Image: https://eyecancer.com/uncategorized/choroidal-metastasis-test/
Loss Minimization View of ML

• Three design decisions
  • Model family: What are the candidate models $f$? (E.g., linear functions)
  • Loss function: How to define “approximating”? (E.g., MSE loss)
  • Optimizer: How do we optimize the loss? (E.g., gradient descent)

• How do we adapt to classification?
Linear Functions for (Binary) Classification

• **Input:** Dataset \( Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \)

• **Regression:**
  • Labels \( y_i \in \mathbb{R} \)
  • Predict \( y_i \approx \beta^T x_i \)

• **Classification:**
  • Labels \( y_i \in \{0, 1\} \)
  • Predict \( y_i \approx 1(\beta^T x_i \geq 0) \)
  • \( 1(C) \) equals 1 if \( C \) is true and 0 if \( C \) is false
  • How to learn \( \beta \)? **Need a loss function!**
Loss Functions for Linear Classifiers

• (In)accuracy:

\[ L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} 1(y_i \neq f_\beta(x_i)) \]

• Computationally intractable

• Often, but not always the “true” loss (e.g., imbalanced data)

\[ L(\beta; Z) = \frac{6}{50} \]
Loss Functions for Linear Classifiers

- **Distance:**

\[
L(\beta; Z) = \frac{1}{n} \sum_{i=1}^{n} \text{dist}(x_i, f_\beta) \cdot 1(f_\beta(x_i) \neq y_i)
\]

- If \( L(\beta; Z) = 0 \), then 100% accuracy
- Variant of this loss results in SVM
- But, we will consider a more general strategy

\[
L(\beta; Z) = 1.2
\]